Radoslaw Krzeski
Summary of Topic VII - Hyperledger

Hyperledger Architecture, Volume 1

I did not find the article mentioned in the seminar topics, so I choose the most credible source I could find that resembles a whitepaper on the technology. The first in a series from Architecture Working Group, and seems to be hosted under the Linux Foundation.

The paper wants to outline something called “permissioned” blockchain networks, something not yet encountered in class, and how this approach enables flexible and interoperable enterprise blockchain technologies.

As mentioned, it lives under the Linux foundation, along and seems to be cooperating with Node.js Cloud Foundry and what is called the Open Container Initiative. This is the infrastructure. The frameworks seem to be built on 5 approaches; Indy, Fabric, Iroha, Sawtooth and Burrow. The tools used are the Hyperledger Composer, Explorer and Cello.

The HAWG exists as a cross-project forum of architects and technologists to promote architectural options and tradeoffs. They aim to create a modular architectural framework for enterprise-class distributed ledgers, building it down to components layers and modules, standardizing interfaces, and interoperability.

As with Corda, business requirements are many. Some require quick consensus approaches, others short block confirmations, or slowing processing time for lower levels of trust. Scalability, confidentially, compliance, workflow complexity and security are others.

Hyperledger takes this and implements distributed ledgers, smart contract engines, client libraries, graphical interface, utility libraries, and sample applications via modular architectural framework. It is a modular approach to a distributed ledger technology. This benefits in terms of the extensibility, flexibility and the ability for components to be modified within the system without affecting other parts.

It is a token-agnostic approach with no native cryptocurrency with the development of a rich use of API`s. There are a few business blockchain components;

- Consensus Layer – generating agreements and confirming correctness of transactions.
- Smart Contract Layer – Processes transaction requests and validating through executing business logic.
- Communication Layer - Responsible for peer-to-peer message transport between nodes in shared ledger.
- Data Store Abstraction - Allows different data-stores to be used by other modules.
- Crypto Abstraction - Allows of crypto algorithms or modules to be swapped out with affecting others.
- Identity Services- Enables establishment of a root of trust, enrolling identities and the management of these.
- Policy services- Policy managements, endorsements, consensus or group management. Module-dependent.
- API`s – Enables clients and applications to interface to blockchains
- Interoperations- Supports interoperation between different blockchain instances.

The document then continues to explore the notion of how consensus work in Hyperledger.

- It must confirm the correctness of transactions, according to endorsements and consensus
- Agrees on order and results of execution
- Interfaces and depends on smart-contract layer to verify correctness of/a set of transactions

It can use algorithms such as Proof of Elapsed Time, Proof of Work or voting, such as the Redundant Byzantine Fault Tolerance and Paxos.

Lottery-based algorithms can scale to many nodes, since the winner proposes a block and transmits it to the rest of the network for validation. Forking is a risk though. Voting-based algorithms provide low-latency finality. When a majority votes, the transactions of a new block passes, but the tradeoff is when many nodes are involved, scalability and speed suffers, but finality is ensured. Networks in Hyperledger operate in partial trust as such.

TABLE 1. COMPARISON OF PERMISSIONED CONSENSUS APPROACHES AND STANDARD PoW

	Permissioned Lottery-based	Permissioned Voting-based	Standard Proof of Work (Bitcoin)
Speed	GOOD	GOOD	POOR
Scalability	GOOD	MODERATE	GOOD
Finality	MODERATE	GOOD	POOR

Consensus is reached in two ways in Hyperledger through two separate activities- Ordering of transactions and validating transactions. By separating these two actions, the Hyperledger allows any implementations of consensus module. Ordering could come from a centralized service to distributed protocols that target different network node models. These can be hashed.

To validate the transactions, it depends on smart contract implement the business logic behind the transaction, and that they conform to the policy given. Invalid transactions are rejected and possibly dropped from the inclusion in a block (being an option of a given implementation?). Two types of truncations errors are given; syntax and logic errors. Logging of these are an option for ledger control. Uses the communication layer to transmit messages between peers.

Consensus must also satisfy two properties: safety and liveness. Safety means that nodes are guaranteed that the same sequences are the same on each node. Liveness means that each non-faulty node will eventually receive every submitted transaction. Consensus is built out differently in each of the five approaches mentioned earlier, shifting between lottery and voting-based methodology. Their names are Kafka in H-Fabric (where the mains procces are endorsements, Ordering and Validation), RBFT in H-Indy, Sumeragi, H-Iroha, PoET in H- Sawtooth

(see document tables for more information)

Overall, the Hyperledger allows for a modular based extendable framework, with commonly defined components and interfaces. This allows for them to be changed independently. It consists of a set of core components and has a shared reference architecture that can be sued by any Hyperledger project.

Hyperledger Fabric Release documents:

States that it is a platform for distributed ledger solutions, underpinned by a modular architecture designed to support pluggable extensions. Apparently, it is uniquely elastic and extensible, planning for the future!

The video mentions that dozens of companies are build a blockchain fabric to support production business networks. It started as a framework to start testing use cases in many areas. They learned that permissions based ledgers that require every peer to execute every transaction, maintain a ledger and run consensus don't scale very well. They can support true privacy however. Fabric was created to remedy this, truly modular, scalable and secure for industrial blockchain solutions. Peers are divided into Endorser, Committer and Consenter.

The video then executes an example of an organic market in California and Radish market in Chile, requiring the transactions to be private in a larger network that support many peers. Fabric covers this privacy with a third-party SDK. The other middle-party curriers can then validate the transaction without necessarily knowing the transaction details that ensued.

Comparison of Ethereum, Hyperledger Fabric and Corda – web article

As I have already outlined many details on both Corda and Ethereum in previous summaries, I will only include new and valuable information these systems that the article might have.

The paper argues that while Corda, also a DLT, is primarily used for financial services, Hyperledger aspires to be used in various industries, extended by the modular base. Ethereum aims to be more general in terms of transactions and applications. The table gives a great comparison:

Table 1

Comparison of Ethereum, Hyperledger Fabric and Corda

Characteristic	Ethereum	Hyperledger Fabric	R3 Corda
Description of platform	– Generic blockchain platform	– Modular blockchain platform	– Specialized distrib-uted ledger platform for financial industry
Governance	– Ethereum developers	– Linux Foundation	– R3
Mode of operation	– Permissionless, public or private[4]	– Permissioned, private	– Permissioned, private
Consensus	– Mining based on proof-of-work (PoW) – Ledger level	– Broad understand-ing of consensus that allows multiple approaches – Transaction level	– Specific understand-ing of consensus (i.e., notary nodes) – Transaction level
Smart contracts	– Smart contract code (e.g., Solidity)	– Smart contract code (e.g., Go, Java)	– Smart contract code (e.g., Kotlin, Java) – Smart legal contract (legal prose)
Currency	– Ether – Tokens via smart contract	– None – Currency and tokens via chaincode	– None

In a DLT, when entities are sharing data and who can contribute, they are called nodes or peers. This created two ways to allow consensus as explained; permissionless and permissioned. If permissionless, anybody is allowed to participate in the network (Ethereum). If permissioned, participants are selected in advance, and access is restricted.

In Ethereum all participants need to reach consensus on the order of transactions (PoW Scheme). Privacy at stake due to the ledger being accessible to everyone, and transactions processing slow to propagate. Corda and Fabric provide a more fine-grained control to record thus enhance privacy. Only parties involved in the transaction must reach consensus.

The paper then gives details on Fabric, the main approach to Hyperledger. It`s consensus approach begins from the proposing of a transaction to the level of committing it to the ledger. Nodes assumes different roles. Here, they are distinguished as clients, peers or orderers.

Client invokes transactions, Peers maintain the ledger to receive updates. Endorsers, a special type of peer, who's task is to endorse transactions, checking if they fulfill necessary conditions. Orderers on the other hand provide a communication channel where messages with transactions can be broadcasted. Channels delivers the same messages to everyone, and at the same logical order to reach consensus. The ledgers also need to be fault tolerant in order to take into order situations where messages are not delivered, although the consensus algorithm is "pluggable" to the extent that it can be swapped. BFT algorithms are mentioned. Consensus only need to be reached at transaction level, and not necessarily ledger level.

Transaction flow: client sends transaction to connected endorsers, update to the ledger. All endorsers agree on update. Client collects all approvals. Approved transaction sent to connected orderers, who then reach their consensus. Then transmitted to peers for committing to ledger.

Two distinction of smart contracts are used: Smart contract code and Smart legal contract. The first is simply software written, and acts as an agent or delegate. Fulfills obligations, rights, and can take control of assets. Executed modelled code. This applies for Fabric in the programming languages of Go or Java. A different word for it is also chaincode. Corda has Ricardian Contract on the other hand, dedicated also for legal prose. Chaincode can be used to build native currency or digital tokens.

Fabric is an expandable toolbox, built for scalability and flexibility, a true DLT.

References:

https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

http://hyperledger-fabric.readthedocs.io/en/release-1.0/

https://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda-21c1bb9442f6